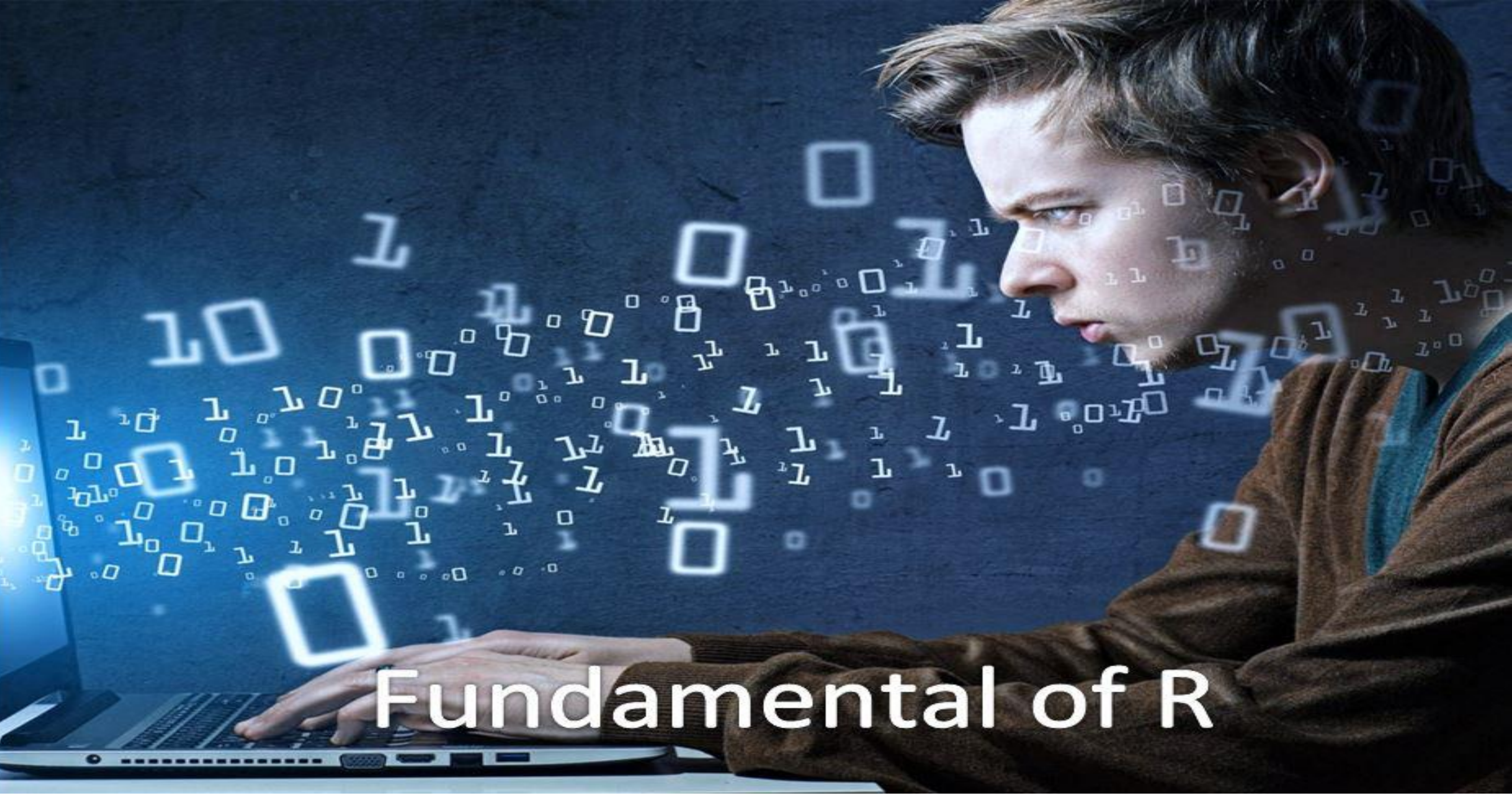


Fundamental of R

Kok Woon Chee
woonchee.kok@gmail.com

UNIMAS
UNIVERSITI MALAYSIA SARAWAK

FoCuS IT
focus on individual transformation

Outline

- Data Objects (Vectors, Matrices, List, Data Frame)
- Reading data from files (format data, clean data)
- Run statistical test

Vectors

- Vector is a fundamental data object in R.
- It is a list of elements
- Any data object in R can be thought of as a vector
- 2 types of vectors: the atomic vector and the lists
- Atomic: contains element of same type nature.
- List: can contain element of any type nature and structure.

Atomic vectors

- In R, use concatenate function (c)
- It can concatenate 3 different types: numerical, character or logical values.

```
>  int_vec <- c(8,10,2,4,6)
>  int_vec
[1] 8 10 2 4 6
> typeof(int_vec)
[1] "double"
> num_vec<-c(3.23,-5.452, 3432.43)
> num_vec
[1]    3.230   -5.452 3432.430
> typeof(num_vec)
[1] "double"
```

Atomic vectors(2)

```
>  char_vec <- c(“apple”, “orange”, “durian”, “mango”)
>  char_vec
[1] “apple” “orange” “durian” “mango”
> typeof(char_vec)
[1] “character"
> log_vec<-c(T, F, F, TRUE, T, T, FALSE)
> log_vec
[1]    TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE
> typeof(log_vec)
[1] “logical"
```

Atomic vectors(3)

- Direct query the nature of the vectors

```
>  is.numeric(int_vec)
[1] TRUE
> Is.character(int_vec)
[1] FALSE
> is.logical(int_vec)
[1] FALSE
```

Atomic vectors(4)

- Convert from type to type

```
>  as.character(int_vec)
[1] “8” “10” “2” “4” “6”
> as.logical(int_vec)
[1] TRUE TRUE TRUE TRUE TRUE
```

Atomic vectors(5)

- Convert from type to type

```
> merged<- c(int_vec, c(num_vec,int_vec))
> merged
 [1]    8.000    10.000    2.000    4.000    6.000    3.230   -5.452
[8]3432.430    8.000    10.000    2.000    4.000    6.000
> merged
 [1]    8.000    10.000    2.000    4.000    6.000    3.230   -5.452
 [8] 3432.430    8.000    10.000    2.000    4.000    6.000
```

Atomic vectors(6)

- Merge vector of different natures, there will be automatic conversion.

```
> c(int_vec, char_vec)
[1] “8” “10” “2” “4” “6” “apple” “orange” “durian”
“mango”
```

Atomic vectors(7)

```
> rnorm(50)
 [1]  2.17489967  1.56436848 -0.41069848 -0.42227040  1.55244378  0.22100349
 [7] -0.31216586  2.33144061  0.06075490 -0.06843641 -0.78797609 -0.05654571
[13]  1.03486873 -0.06214879  1.28016996 -0.27971040  0.24613821  0.66722639
[19]  1.33769367  1.39769777  1.63967540  0.52847935 -0.27981692 -0.45326377
[25]  0.15047811  0.92136397  1.14991204  0.62921306 -0.75841810 -1.39687149
[31] -0.82832679 -0.17146222  0.27955510  0.06332471 -0.43336223  1.31862156
[37]  0.32166745  0.09309871  0.02418356  0.78641048  0.36777002 -0.72566463
[43]  0.25648912 -0.43113136  0.80751969  0.65995328 -0.33727980 -0.78240453
[49]  1.50634413  0.08457504
```

Atomic vectors(8)

```
> ls()
 [1] "a"        "b"        "c"        "d"        "e"        "figure"
 [7] "hfmd"     "mLL"      "mLL2"     "model"    "my_int"   "my_var"
[13] "Patients" "Week"
> ls()->content
> content
 [1] "a"        "b"        "c"        "d"        "e"        "figure"
 [7] "hfmd"     "mLL"      "mLL2"     "model"    "my_int"   "my_var"
[13] "Patients" "Week"
> typeof(content)
[1] "character"
> is.character(content)
[1] TRUE
```

Metadata

- Data about data=information about the data

```
> attributes(int_vec)
NULL
> names(int_vec)
NULL
> names(int_vec)<-c("a", "b", "c", "d", "e")
> names(int_vec)
[1] "a" "b" "c" "d" "e“
> attributes(int_vec)
$names
[1] "a" "b" "c" "d" "e"
```

- Define the name alternatively

```
> int_vec2<-c(A=3,B=5,C=3,G=8)
> int_vec2
A B C G
3 5 3 8
```

- Remove the names

```
> unname(int_vec2)
[1] 3 5 3 8
> int_vec2
A B C G
3 5 3 8
```

Factor for categorical variables

- Factor is qualitative variable.

```
> gender<-c(4,5,4,4,4,5,5,4,5,5)
> gender
 [1] 4 5 4 4 4 5 5 4 5 5
> typeof(gender)
[1] "double“
> attributes(gender)
NULL
> as.factor(gender)->gender
> gender
 [1] 4 5 4 4 4 5 5 4 5 5
Levels: 4 5
```

Factor for categorical variables (cont.)

```
> attributes(gender)
$levels
[1] "4" "5"
$class
[1] "factor"
> levels(gender)
[1] "4" "5“
> levels(gender)<-c("male", "female")
> gender
[1] male   female male   male   male   female female male   female
[10]female
Levels: male female
> as.numeric(gender)
 [1] 1 2 1 1 1 2 2 1 2 2
> as.character(gender)
[1] "male"   "female" "male"   "male"   "male"   "female" "female"
[8]"male" "female" "female"
```

More Attributes

- Define more attributes that can be define

```
> int_vec
 a  b  c  d  e 
 8 10  2  4  6 
> attributes(int_vec)
$names
[1] "a" "b" "c" "d" "e"
> attr(int_vec,"source")<-"www.ecdc.europa.eu"
> attributes(int_vec)
$names
[1] "a" "b" "c" "d" "e"
$source
[1] www.ecdc.europa.eu
> int_vec
 a  b  c  d  e 
 8 10  2  4  6 
```

- Retrieve any specific attribute with the same function

```
> attributes(int_vec)
$names
[1] "a" "b" "c" "d" "e"
$source
[1] "www.ecdc.europa.eu"
> names(int_vec)
[1] "a" "b" "c" "d" "e"
> attr(int_vec,"name")
[1] "a" "b" "c" "d" "e"
> attr(int_vec,"source")
[1] "www.ecdc.europa.eu"
```

Retrieving elements

- By specify the list of indexes of the elements
- By specify the list of the names of the elements (provided that names are defined)
- By specify for each element of the object whether want to select it or not with a logical value

Retrieving elements (2)

```
> int_vec
 a  b  c  d  e
 8 10  2  4  6
#first method
> int_vec[c(1,3)]
a c
8 2
#second method
> int_vec[c("a","c")]
a c
8 2
#third method
> int_vec[c(T,F,T,F,F)]
a c
8 2
```

Remove elements

```
> int_vec
 a  b  c  d  e
 8 10  2  4  6
#first method
> int_vec[-c(1,3)]
 b  d  e
10  4  6
#third method
> int_vec[!c(T,F,T,F,F)]
 b  d  e
10  4  6
```

Selection criteria

```
> int_vec
 a  b  c  d  e
 8 10  2  4  6
> sel<-int_vec>5
> sel
    a     b     c     d     e
 TRUE  TRUE FALSE FALSE  TRUE
> int_vec[sel]
 a  b  e
 8 10  6
#can be done in single line
> int_vec[int_vec>5]
 a  b  e
 8 10  6
```

Matrices

- Matrix is a vector with dimension attributes by giving number of rows and columns.

```
> vec<-c(3,2,5,5,4,7,7,9,7,5,3,1)
> vec
 [1] 3 2 5 5 4 7 7 9 7 5 3 1
> attributes(vec)
NULL
> mat<-matrix(vec,ncol=4)
> mat
     [,1] [,2] [,3] [,4]
[1,]    3    5    7    5
[2,]    2    4    9    3
[3,]    5    7    7    1
> attributes(mat)
$dim
[1] 3 4
```

Matrices (2)

- Matrix is internally a vector, can access the matrix like vector

```
> typeof(mat)
[1] "double"
> length(mat)
[1] 12
> typeof(vec)
[1] "double"
> length(vec)
[1] 12
```

Matrices (3)

- Matrix is internally a vector, can access the matrix like vector

```
> mat[c(1,2,11,12)]
[1] 3 2 3 1
> mat[-c(1,2,11,12)]
[1] 5 5 4 7 7 9 7 5
> mat[mat<4]
[1] 3 2 3 1
> mat[!mat<4]
[1] 5 5 4 7 7 9 7 5
```

Matrices (4)

- Matrix is internally a vector, can access the matrix like vector

```
> dim(mat)
[1] 3 4
> mat[2,3]
[1] 9
> mat[c(2,3),c(3,4)]
     [,1] [,2]
[1,]    9    3
[2,]    7    1
> mat[2,1]
[1] 2
> mat[2,]
[1] 2 4 9 3
> mat[, c(3,4)]
     [,1] [,2]
[1,]    7    5
[2,]    9    3
[3,]    7    1
```

Matrices (5)

- Matrix is internally a vector, can access the matrix like vector

```
> mat[-2,-3]
     [,1] [,2] [,3]
[1,]    3    5    5
[2,]    5    7    1
> mat[-c(2,3),-c(3,4)]
[1] 3 5
> mat[-2,]
     [,1] [,2] [,3] [,4]
[1,]    3    5    7    5
[2,]    5    7    7    1
> mat[,-c(3,4)]
     [,1] [,2]
[1,]    3    5
[2,]    2    4
[3,]    5    7
```

Matrices (6)

- To retrieve the element in 2-dimension of the matrix, we specify the row and column indexes inside the [] bracket.
- If want all elements in matrix, we leave the space blank.
- We can use logical criteria to retrieve elements of a matrix.

```
> mat[c(F,T,F), c(F,F,T,F)]
[1] 9
> mat[!c(F,T,T),!c(F,F,T,T)]
[1] 3 5
```

Matrices (7)

- Define the name for matrix elements

```
> mat
    [,1] [,2] [,3] [,4]
[1,]    3    5    7    5
[2,]    2    4    9    3
[3,]    5    7    7    1
> attributes(mat)
$dim
[1] 3 4

> rownames(mat)<-c("A","B","C")
> colnames(mat)<-c("a","b","c","d")
> mat
  a b c d
A 3 5 7 5
B 2 4 9 3
C 5 7 7 1
```

Matrices (8)

- Using defined name to retrieved the elements.

```
> mat["A",]
a b c d
3 5 7 5


> mat["A",c("c","d")]
c d
7 5
```

List (1)

- List can contain
  - Element of various nature and structure (including lists)
  - Hierarchical structure (whereas atomic vectors have a flat structure).

```
#make list
> list(mat,vec,int_vec,num_vec,char_vec,log_vec) -> my_list
#return the length
> length(my_list)
#return the structure
>str(my_list)
```

List (2)

- Check the name of the list items.

```
> names(my_list)
NULL
> names(my_list)<-c("A","B","C","D","E","F")
#check structure again
> str(my_list)
```

List (3)

- Names can also be defined at creation time.

```
> my_list2<-list(the_mat=mat,the_vec=vec)
> my_list2
$the_mat
  a b c d
A 3 5 7 5
B 2 4 9 3
C 5 7 7 1

$the_vec
 [1] 3 2 5 5 4 7 7 9 7 5 3 1
```

List (4)

- Elements can be accessed in the same way as for the atomic vectors.

```
#by the index
> my_list[2]
#by using the names
> my_list[c("A","C")]
#remove the elements
> my_list[-c(4,5)]
#by using logical values
>my_list[c(T,T,F,T,F,F)]
```

List (5)

- Single [] symbol maintains the list structure.
- To remove it, use double [[]] or dollar sign($).

```
> my_list[2]
$B
 [1] 3 2 5 5 4 7 7 9 7 5 3 1

> my_list[[2]]
 [1] 3 2 5 5 4 7 7 9 7 5 3 1

> my_list$B
 [1] 3 2 5 5 4 7 7 9 7 5 3 1
```

List (6)

- Element can be added to a list this way

```
> c(my_list2,c(3,4))
$the_mat
  a b c d
A 3 5 7 5
B 2 4 9 3
C 5 7 7 1

$the_vec
 [1] 3 2 5 5 4 7 7 9 7 5 3 1

[[3]]
[1] 3

[[4]]
[1] 4
```

List (7)

- Remove the list structure and transform it into a vector

```
> unlist(my_list2)
 the_mat1  the_mat2  the_mat3  the_mat4  the_mat5
the_mat6  the_mat7  the_mat8  the_mat9 the_mat10
        3         2         5         5         4         7         7         9
7         5
the_mat11 the_mat12  the_vec1  the_vec2  the_vec3
the_vec4  the_vec5  the_vec6  the_vec7  the_vec8
        3         1         3         2         5         5         4         7
7         9
 the_vec9 the_vec10 the_vec11 the_vec12
        7         5         3         1
```

Data Frame

- Data frames are the list:
  - Element are atomic vectors of the same length,
  - but not necessarily of the same nature.

Data Frame (2)

- Define data frame

```
>  items<-c("orange", "apple", "pineapple", "mango", "banana")
> price<-c(3.45,7.12,12.32,5,7)
>  site<-c("Kuching", "Kota Samarahan", "Kota Samarahan",
“Kuching", “Kuching")
> market<-c(T,F,T,F,F)
> groceries<-data.frame(items, price,site,market)
> groceries
      items   price           site      market
1    orange    3.45        Kuching        TRUE
2     apple    7.12 Kota Samarahan       FALSE
3 pineapple   12.32 Kota Samarahan        TRUE
4     mango    5.00        Kuching       FALSE
5    banana    7.00        Kuching       FALSE
```

Data Frame (3)

- Check the data structure
- Return the summary of each data

```
> str(groceries)


> summary(groceries)
```

Data Frame (4)

- The row are observations and the columns are variables.
- The character variables are by default considered as factors.
- Since they are lists, their elements can be accessed in the way as for lists.

```
> names(groceries)
[1] "items"  "price"  "site"   "market"
> groceries[1]
      items
1    orange
2     apple
3 pineapple
4     mango
5    banana
```

Data Frame (5)

```
> groceries[-c(2,4)]
      items               site
1    orange            Kuching
2     apple     Kota Samarahan
3 pineapple     Kota Samarahan
4     mango            Kuching
5    banana            Kuching

> groceries["market"]
  market
1   TRUE
2  FALSE
3   TRUE
4  FALSE
5  FALSE
```

Data Frame (6)

```
> groceries[c(T,F,T,F)]
      items                 site
1    orange              Kuching
2     apple       Kota Samarahan
3 pineapple       Kota Samarahan
4    mango               Kuching
5    banana              Kuching


> groceries$items
[1] orange   apple    pineapple mango
banana
Levels: apple banana mango orange pineapple
```

Data Frame (7)

- Data frame is a 2 x 2 structure, therefore can accessed elements the same way for matrices.

```
> groceries[1:2,3:4]

> groceries[,-2]

> groceries[-1,]

> groceries[-c(T,T,F,F,F),c(T,F,T,F)]

>groceries[, “site”]
```

Data Frame (8)

- Managing the data frame

```
#return the item that can find in the market
> groceries[groceries$market,]
        items        price          site    market
1      orange        3.45       Kuching       TRUE
3     pineapple        12.32 Kota Samarahan   TRUE
#sorting the data frame according to price
> groceries[order(groceries$price),]
     items price         site market
1    orange  3.45        Kuching   TRUE
4     mango  5.00        Kuching  FALSE
5    banana  7.00        Kuching  FALSE
2     apple  7.12 Kota Samarahan  FALSE
3 pineapple 12.32 Kota Samarahan   TRUE
```

Data Frame (8)

- Managing the data frame

```
#return the most expensive item and where is it?
> groceries[groceries$price ==
max(groceries$price), c("items","site")]
      items         site
3 pineapple Kota Samarahan
```

Reading data from files

- Data sets are most of the time in excel spreadsheet of equivalent.
- Sometimes they are in a database system
- R can access both database systems and excel files.
- Basic way to read data from a simple text files.
- Can Copy & Paste the content of excel file sheet to a simple text file,

Read data

```
> read.table("gr2.txt",header=T,sep="\t")
       item   price              location  market
1     apple       4               Kuching    TRUE
2    orange       6               Kuching    TRUE
3      susu       3        Kota Samarahan   FALSE
4 pineapple       5        Kota Samarahan    TRUE
5    papaya       5               Kuching   FALSE
#alternative to read data
> read.table(file.choose(),header=T,sep="\t")
> read.delim(“gr2.txt”)
```

Putting data into good format

```
>data1<-read.csv(file.choose(),sep=",", head=TRUE)

>head(data1)

>tail(data1)
```

Run a statistical test

- Basic Statistic

Basic stats	**R syntax**
Mean	mean(variable_name)
Median	median(variable_name)
The largest value in the variable	max(variable_name)
The smallest value in the variable	min(variable_name)
The standard deviation	sd(variable_name)
The number of items in the variable	length(variable_name)
The variance	var(variable_name)
Any quantile (level can be 0.25,0.75 others)	Quantile(variable_name, level)

Run a statistical test

- Read the data and check the structure

```
> data1<-read.csv(file.choose(),sep=",", head=TRUE)
> str(data1)
```

Run a statistical test

- Data Introduction

English Language Arts (ELA) Test Results from year 2006 until 2012 in New York City. The data are disaggregated by student ethnicity. With the adoption of the New York Common Core Learning Standards (CCLS) in ELA/ Literacy and Mathematics, students in grades 3 until 8 required to take ELA test each spring. Educators use ELA test result to assign students to appropriate classes and identify areas where the student needs extra help or more challenging material. At the same time, teachers and principals use the results to make decisions about the promotion and summer school. Nevertheless, educators also examine school-wide results to identify broad instructional areas that require improvement. Different level will be assigned to student based on their marks (Test Result, 2012). There are Level 1, Level 2, Level 3 and Level 4. Level 1 indicates lowest proficiency level or known as well below proficient while Level 4 shows advanced level.

Run a statistical test (2)

- Summarize the data set in efficient way

```
> table(data1$Grade,data1$Demographic)

            Asian Black Hispanic White
 3              7     7        7     7
 4              7     7        7     7
 5              7     7        7     7
 6              7     7        7     7
 7              7     7        7     7
 8              7     7        7     7
 All Grades     7     7        7     7
```

Run a statistical test (3)

- Basic statistic

```
summary(data1)
> sd(data1$Mean.Scale.Score)
[1] 12.66696
> quantile(data1$Mean.Scale.Score)
0%  25%  50%  75% 100%
630  654  663  672  686
```

Run a statistical test (3)

- Questions

1. Is there any relationship between the numbers of student who take ELA and the mean scale score?

2. Which ethnic perform better in ELA test in mean scale score?

Run a statistical test (4)

1. Average/mean of Mean Scale Score in the data is 661.8.
2. Median of Mean Scale Score in the data is 663.0
3. The standard deviation of Mean Scale Score in the data is 12.66696.
4. The minimum value of Mean Scale Score is 630 and maximum value is 686.
5. First quartile (Q_1) is 654 and third quartile of Mean Scale Score (Q_3) is 672.
6. The Interquartile Range (IQR) of Mean Scale Score is 18.

Run a statistical test (5)

- Find the relationship between the number of students and the mean scale score.

```
 >reg1<-lm(data1$Mean.Scale.Score ~ data1$Number.Tested)


>plot(data1$Number.Tested,data1$Mean.Scale.Score,
main="Number of Students take ELA test with the Mean Scale Score
from 2006-2011", xlab="Number of Tested", ylab="Mean Scale
Score")


>abline(reg1)
#find correlation
>cor(data1$Number.Tested,data1$Mean.Scale.Score)
```

Run a statistical test (5)

- Find the relationship between the mean scale score and the ethnic.

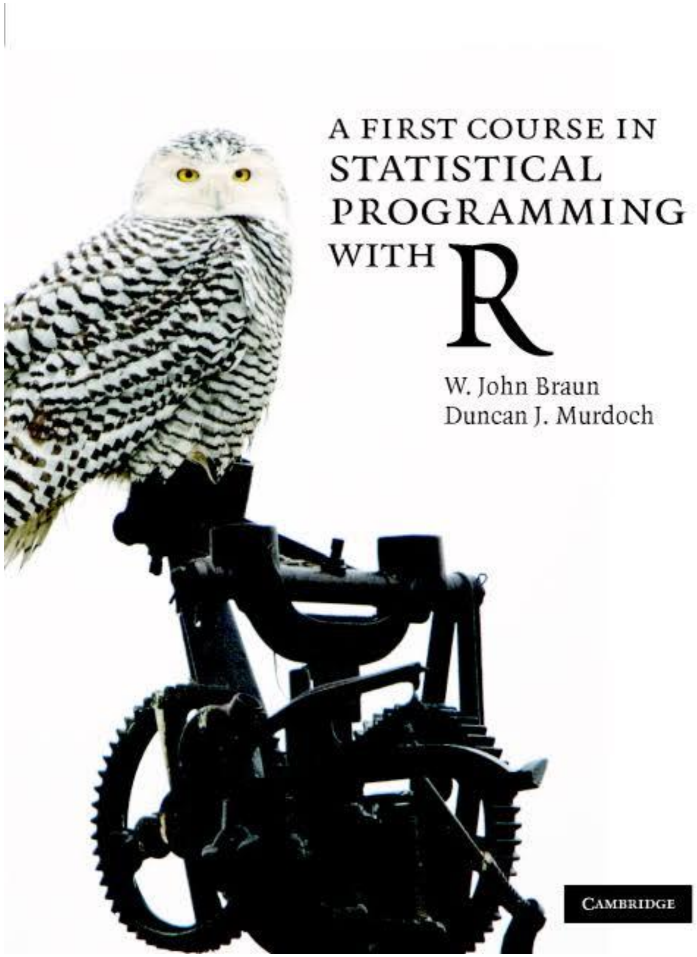
```
> boxplot(data1$Mean.Scale.Score~data1$Demographic)
```

Run a statistical test (6)

- Sub-setting the ethnic group and see the summary of each group.

```
> black<-subset(data1,Demographic=="Black")
> hispanic<-
subset(data1,Demographic=="Hispanic")
> white<-
subset(data1,Demographic=="White")
> asian<-subset(data1,Demographic=="Asian")
>summary (black)
```

References



Data retrieved from: https://data.cityofnewyork.us/Education/English-Language-Arts-ELA-Test-Results-2006-2012-C/p5w7-g72z

THANK
YOU